What is claimed is:

1. A method for operating a computer using object-based computer code of an object-oriented programming language, the method comprising:

utilizing an explicit interface member mechanism that enables at least one software component to implement at least one explicit interface member by explicitly specifying the relationship between said at least one software component and the at least one interface member.

2. A method according to claim 1, wherein said specifying of the relationship includes specifying a qualified name of the at least one software component.

3. A method according to claim 2, wherein said specifying of the qualified name includes specifying at least one interface name and said at least one interface member name.

4. A method according to claim 1, wherein said explicit interface member mechanism enables an explicit interface member implementation to be excluded from the public interface of said at least one software component.

5. A method according to claim 1, wherein the at least one software component is at least one of a class or struct instance, as defined by the object-oriented programming language.

6. A method according to claim 1, wherein the explicit interface member mechanism enables said at least one software component to implement an internal interface not accessible to a consumer of said at least one software component.

7. A method according to claim 1, wherein said explicit interface member mechanism enables disambiguation of a plurality of interface members having the same signature.

8. A method according to claim 1, wherein said explicit member mechanism enables disambiguation of a plurality of interface members having the same signature and return type.

9. A method according to claim 1, wherein in addition to allowing the implementation of public interface members, said explicit interface member mechanism enables the implementation of private interface members.

10. A method according to claim 1, wherein said explicit interface member mechanism enables the implementation of a plurality of non-conflicting specific versions of a generic interface.

11. A method according to claim 1, wherein the computer code is programmed according to an object-oriented programming language, and said object-oriented programming language is one of C#, Fortran, Pascal, Visual Basic, C, C++ and Java.

12. A method according to claim 1, wherein an implementation of an explicit interface member is a method, property, event, or indexer declaration that references a fully qualified interface member name.

13. A method according to claim 1, wherein said at least one software component names an interface in the base class list of the at least one software component that contains a member whose fully qualified name, type, and parameter types exactly match those of the implementation of the explicit interface member.

14. A method according to claim 1, wherein said explicit interface member mechanism includes an interface mapping mechanism that locates implementations of interface members in said at least one software component.

15. A method according to claim 14, wherein said interface mapping mechanism locates an implementation for each member of each interface specified in the base class list of the at least one software component.

16. A method according to claim 1, wherein said at least one software component inherits

all interface implementations provided by its base classes

17. A method according to claim 1, wherein it is not possible to override an explicit interface member implementation, but where an explicit interface member implementation calls another virtual method, derived classes are capable of overriding the implementation.

18. A method according to claim 1, wherein a software component of said at least one software component that inherits an interface implementation is permitted to re-implement the interface by including the interface in the base class list of the software component.

19. A method according to claim 1, wherein said explicit interface member mechanism prevents conflict among specific implementations of a generic interface.

20. A computer readable medium bearing computer executable instructions for carrying out the method of claim 1.

21. A modulated data signal carrying computer executable instructions for performing the method of claim 1.

22. A computing device comprising means for performing the method of claim 1.

23. A computer readable medium having stored thereon a plurality of computer-executable modules written in an object-oriented programming language, the computer executable modules comprising:

an explicit interface member mechanism that enables at least one software component to implement at least one explicit interface member by explicitly specifying the relationship between said at least one software component and the at least one interface member.

24. A computer readable medium according to claim 23, wherein said specifying of the relationship includes specifying a qualified name of the at least one software component.

25. A computer readable medium according to claim 24, wherein said specifying of the qualified name includes specifying at least one interface name and said at least one interface member name.

26. A computer readable medium according to claim 23, wherein said explicit interface member mechanism enables an explicit interface member implementation to be excluded from the public interface of said at least one software component.

27. A computer readable medium according to claim 23, wherein the at least one software component is at least one of a class or struct instance, as defined by the object-oriented programming language.

28. A computer readable medium according to claim 23, wherein the explicit interface member mechanism enables said at least one software component to implement an internal interface not accessible to a consumer of said at least one software component.

29. A computer readable medium according to claim 23, wherein said explicit interface member mechanism enables disambiguation of a plurality of interface members having the same signature.

30. A computer readable medium according to claim 23, wherein said explicit member mechanism enables disambiguation of a plurality of interface members having the same signature and return type.

31. A computer readable medium according to claim 23, wherein in addition to allowing the implementation of public interface members, said explicit interface member mechanism enables the implementation of private interface members.

32. A computer readable medium according to claim 23, wherein said explicit interface member mechanism enables the implementation of a plurality of non-conflicting specific versions of a generic interface.

33. A computer readable medium according to claim 23, wherein the object-oriented programming language is one of C#, Fortran, Pascal, Visual Basic, C, C++ and Java.

34. A computer readable medium according to claim 23, wherein an implementation of an explicit interface member is a method, property, event, or indexer declaration that references a fully qualified interface member name.

35. A computer readable medium according to claim 23, wherein said at least one software component names an interface in the base class list of the at least one software component that contains a member whose fully qualified name, type, and parameter types exactly match those of the implementation of the explicit interface member.

36. A computer readable medium according to claim 23, wherein said explicit interface member mechanism includes an interface mapping mechanism that locates implementations of interface members in said at least one software component.

37. A computer readable medium according to claim 36, wherein said interface mapping mechanism locates an implementation for each member of each interface specified in the base class list of the at least one software component.

38. A computer readable medium according to claim 23, wherein said at least one software component inherits all interface implementations provided by its base classes

39. A computer readable medium according to claim 23, wherein it is not possible to override an explicit interface member implementation, but where an explicit interface member implementation calls another virtual method, derived classes are enabled to override the implementation.

40. A computer readable medium according to claim 23, wherein a software component of said at least one software component that inherits an interface implementation is permitted

to re-implement the interface by including it in the base class list of the software component.

41. A computer readable medium according to claim 23, wherein said explicit interface member mechanism prevents conflict among specific implementations of a generic interface.

42. An object-oriented programming language for producing computer executable modules, comprising:

an explicit interface member mechanism that enables at least one software component to implement at least one explicit interface member by explicitly specifying the relationship between said at least one software component and the at least one interface member.

43. An object-oriented programming language according to claim 42, wherein said specifying of the relationship includes specifying a qualified name of the at least one software component.

44. An object-oriented programming language according to claim 43, wherein said specifying of the qualified name includes specifying at least one interface name and said at least one interface member name.

45. An object-oriented programming language according to claim 42, wherein said explicit interface member mechanism enables an explicit interface member implementation to be excluded from the public interface of said at least one software component.

46. An object-oriented programming language according to claim 42, wherein the at least one software component is at least one of a class or struct instance, as defined by the object-oriented programming language.

47. An object-oriented programming language according to claim 42, wherein the explicit interface member mechanism enables said at least one software component to implement an internal interface not accessible to a consumer of said at least one software component.

48. An object-oriented programming language according to claim 42, wherein said explicit interface member mechanism enables disambiguation of a plurality of interface members having the same signature.

49. An object-oriented programming language according to claim 42, wherein said explicit member mechanism enables disambiguation of a plurality of interface members having the same signature and return type.

50. An object-oriented programming language according to claim 42, wherein in addition to allowing the implementation of public interface members, said explicit interface member mechanism enables the implementation of private interface members.

51. An object-oriented programming language according to claim 42, wherein said explicit interface member mechanism enables the implementation of a plurality of non-conflicting specific versions of a generic interface.

52. An object-oriented programming language according to claim 42, wherein the object-oriented programming language is one of C#, Fortran, Pascal, Visual Basic, C, C++ and Java.

53. An object-oriented programming language according to claim 42, wherein an implementation of an explicit interface member is a method, property, event, or indexer declaration that references a fully qualified interface member name.

54. An object-oriented programming language according to claim 42, wherein said at least one software component names an interface in the base class list of the at least one software component that contains a member whose fully qualified name, type, and parameter types exactly match those of the implementation of the explicit interface member.

55. An object-oriented programming language according to claim 54, wherein said explicit interface member mechanism includes an interface mapping mechanism that locates implementations of interface members in said at least one software component.

56. An object-oriented programming language according to claim 42, wherein said interface mapping mechanism locates an implementation for each member of each interface specified in the base class list of the at least one software component.

57. An object-oriented programming language according to claim 42, wherein said at least one software component inherits all interface implementations provided by its base classes

58. An object-oriented programming language according to claim 42, wherein it is not possible to override an explicit interface member implementation, but where an explicit interface member implementation calls another virtual method, derived classes are enabled to override the implementation.

59. An object-oriented programming language according to claim 42, wherein a software component of said at least one software component that inherits an interface implementation is permitted to re-implement the interface by including it in the base class list of the software component.

60. An object-oriented programming language according to claim 42, wherein said explicit interface member mechanism prevents conflict among specific implementations of a generic interface.